

引用格式: 崔丹丹, 宫永生. NVMe 高速存储的 FPGA 实现 [J]. 微电子学与计算机, 2019, 36(6): 26-30. [CUI D D, GONG Y S. A high-speed storage system of NVMe based on FPGA [J]. Microelectronics & Computer, 2019, 36(6): 26-30.]

NVMe 高速存储的 FPGA 实现

崔丹丹^{1,2}, 宫永生²

(1 中国科学院大学, 北京 101400; 2 中国科学院太空应用重点实验室,
中国科学院 空间应用工程与技术中心, 北京 100094)

摘 要: 针对当前 SATA 接口数据传输低延迟和读写性能的不足, 结合空间应用对主控系统的需求, 进行了高速接口设计, 本文提出了一种 NVMe 高速存储的 FPGA 实现方案. 基于 Xilinx 公司的 ZC706 评估板, 参考 NVMe 协议 1.3a, 运用 VHDL 硬件描述语言, 采用模块化设计理念, 完成 NVMe 主控 IP 核的设计. 经仿真验证表明, 该主控 IP 核能正确完成读、写、设备复位、识别等操作, 可应用于嵌入式存储系统.

关键词: NVMe1.3a; 主控端; 高速; FPGA; VHDL

中图分类号: TP3

文献标识码: A

文章编号: 1000-7180(2019)06-0026-05

A high-speed storage system of NVMe based on FPGA

CUI Dan-dan^{1,2}, GONG Yong-sheng²

(1 University of Chinese Academy of Science, Beijing 101400, China; 2 Key Laboratory of Space Utilization, Technology and Engineering Center for Space Utilization, Chinese Academy of Science, Beijing 100094, China)

Abstract: In view of the shortcomings of low-latency data transmission and fast reading and writing of the current SATA interface, combined with the requirement of space application to the host controller, this paper proposed a FPGA implementation scheme for NVMe high-speed storage. It takes NVMe 1.3a protocol for reference and is implemented on the basis of ZC706 evaluation board of Xilinx with modular design concept in VHDL language. According to the simulation verification, it shows that the controller IP core can realize many operations, such as reading, writing, device reset and identification. It can be applied to embedded storage system.

Key words: NVMe1.3a; Host; High-speed; FPGA; VHDL

1 引言

利用固态电子存储芯片阵列(NAND Flash)制成的固态硬盘(SSD)已经成为数据储存的主要途径,传统 SSD 大多采用 SATA 接口,运行 AHCI/SATA 协议,而 SATA3.0 理论最大传输速率为 600 MB/s^[1](实际速度为 560 MB/s).随着我国空间站的建设,整个空间系统的数据网将变得更加复杂、庞大,解决数据传输的实时性、提高功耗的管控能力等迫在眉睫,SATA 接口的极限传输速率逐渐

无法满足空间应用领域的存储性能需求,已经成为高带宽、大容量数据存储系统的发展瓶颈.

基于 PCIe 接口的 SSD 加上非易失性存储器协议(Non-Volatile Memory Express, NVMe)可以有效解决接口的性能问题. NVMe 是闪存类存储介质的逻辑接口总线传输协议规范,此规范专为 SSD 而设计,规定了主机与 SSD 之间的通信方式,具有精简的内存接口、命令集和队列设计^[2],其底层为 PCIe 通道,充分利用了 PCIe 的低延时和并行性,在可控制的存储成本下,极大的提升了数据传输带宽

和指令并行处理能力,彻底解放了 SATA 时代固态硬盘的极致性能.在航天应用领域,FPGA 以其丰富的逻辑资源与可靠性保证一直被作为主控系统的首选方案.因此本文将结合 FPGA 进行 NVMe 接口设计,保证数据高速传输,以适应未来空间领域中大容量、高宽带的存储需求.

2 NVMe 简介

NVMe 作为针对非易失性固态存储架构优化的标准协议,定义了最优化的寄存器接口、指令集和功能集,在设计时充分考虑了存储介质和传输过程,优化数据传输路径和传输方式,极大提高了数据传输效率,降低了数据访问延迟.

2.1 PCIe 接口

NVMe 物理层采用 PCIe 通道进行数据传输,PCIe 是一种全双工的串行总线,具有带宽可扩展性的特性^[3],总线版本与吞吐量的关系如表 1 所示^[4-5].由此可见,其速率远高于 SATA3.0 的理论最大传输速率.

表 1 总线版本与吞吐量的关系(单工速度)

PCIe 版本	编码方式	Lane-吞吐量(MB/s)		
		×1	×4	×16
1.0	8b/10b	250	1 000	4 000
2.0	8b/10b	500	2 000	8 000
3.0	128b/130b	985	1 969	15 754
4.0	128b/130b	1 969	3 938	31 508

作为 NVMe 协议的底层通信接口,PCIe 总线规范采用了分层的体系结构,依次为事务层、数据链路层和物理层,其中事务层作为 NVMe 协议与 PCIe 接口的信息交互层,主要负责接收主控端生成的 TLP 包并发送到数据链路层,发送包含设备端数据和命令的 TLP 包至主控端,因此根据 NVMe 命令的执行过程,每次读写操作会经过多次 PCIe TLP 包的信息交互.

一个 TLP 包含多个可选的前缀、头、数据载荷和可选的后缀组成,其中头是 TLP 最重要的标识,包含当前 TLP 的总线事务类型、路由信息、有无后缀等一系列信息.当存储器读写 TLP 支持 64 bit 地址模式时,头长度为 4 DW,否则为 3 DW.数据载荷的长度可变,最小为 0,最大为 1 024 DW.

2.2 NVMe 队列

NVMe 协议主要基于提交队列 SQ 和完成队列 CQ,二者均分配于主控端存储空间中.当主控端发

送命令时,提交队列完成命令的存放与发送;当 SSD 执行完命令后,会返回相应的执行状态,放入完成队列中.

根据 NVMe 命令的不同,提交队列和完成队列分为 Admin、I/O 两种. Admin 队列用于存放 Admin 命令码(如设备识别、创建和删除 I/O 命令的 SQ 和 CQ 等),I/O 队列是 NVMe 协议的核心,包含读、写等相关信息(如逻辑块数量 NLB 和读写的起始地址 SLBA 等).

队列为固定大小的环形缓冲区,采用生产-消耗模型,规定队尾是生产者,指向队列中的下一个空位,队首是消耗者,指向下一个将要被执行命令所在的位置.

NVMe 协议对 Admin SQ/CQ 和 IO SQ/CQ 有不同的约定,如下所示:

(1)系统有且仅有一对 Admin SQ/CQ,但可以有最多 64 K 对 I/O SQ/CQ;

(2)Admin SQ/CQ 的队列深度可达 4 096 (4K);而 I/O SQ/CQ 的队列深度可达 65 536 (64K);

(3)Admin、I/O SQ 条目大小为 64 B,对应的 CQ 条目大小为 16 B;

(4)Admin SQ 和 CQ 是一对一的,而 I/O SQ 和 CQ 可以一对一或多对一,多个 SQ 可以支持多线程工作,不同 SQ 之间可以赋予不同的优先级;

(5)Admin 和 I/O 的 SQ、CQ 均放在主控端内存中,但 SQ 由主控端更新,CQ 则由设备端更新.

2.3 NVMe 命令

NVMe 是一种主控端与设备端之间的通讯协议,在协议栈中隶属高层,即命令层,它所发出的命令均通过 PCIe 总线去完成. NVMe 有两种命令: admin 命令和 I/O 命令. admin 命令用以主控端管理和控制设备端,而 I/O 命令用于二者之间的数据传输.

命令的传输除了需要 SQ、CQ 外,还需要门铃寄存器 DB 以及 PCIe 接口.每个 SQ 和 CQ 都有两个对应的 DB: Head DB 和 Tail DB,DB 是 SSD 端的寄存器,记录 SQ 和 CQ 的头和尾的位置,即读出和写入指针.命令的执行过程主要分为 8 步,如图 1 所示:

(1)主控端在自身内存中构建 SQ 和对应的 CQ,并将命令写入 SQ 尾部;

(2)主控端更新 SQ 尾指针 DB 寄存器,通知设备端到主控端提取新命令;

- (3)设备端查询 DB 的更新值,到主控端 SQ 头部提取命令,并实时更新设备端内部 SQ 头部指针;
- (4)设备端解析并执行命令,进行相应的读写操作;
- (5)设备端完成一条指令后,将执行结果封装为完成包并写入主控端 CQ 中;
- (6)设备端向主控端发送一个与回执信息相对应的中断信息,通知主控端命令执行完毕;
- (7)主控端收到中断后,解析 CQ 中的命令完成包并进行相应处理;
- (8)主控端处理完毕后,释放完成队列位置,并更新 CQ 头指针 DB 寄存器。

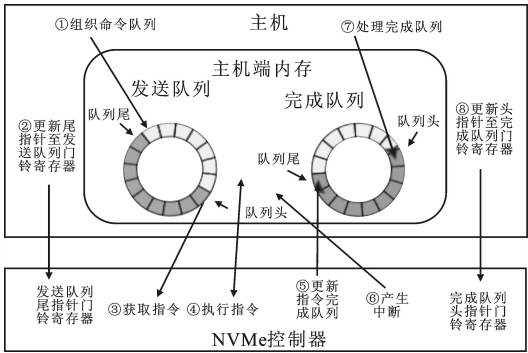


图 1 NVMe 命令处理流程

相较于 SATA/AHCI 协议,NVMe 协议指令执行交互过程简单,队列数量和深度远胜于前两者,多个队列指令并行操作及简单的指令发送机制保证了读写操作的高效执行^[6-7]。

2.4 NVMe 寄存器

NVMe 协议利用 PCIe 设备基本配置空间中基地址寄存器 0 和 1 所指示的空间,定义了一组控制寄存器,用于初始化以及指令执行中指针位置的更新,主要包括:

- (1)CAP 寄存器:指示 SSD 可支持的主控端内存页大小范围、命令集、DB 寄存器地址间隔、仲裁方式、I/O SQ 最大队列深度等;
- (2)CC 寄存器:配置 I/O SQ 和 CQ 的位宽、关机提醒、仲裁方式、主控端内存页大小、控制器使能等;
- (3)CSTS 寄存器:指示当前的关机进度、控制器错误信息以及是否准备完成等;
- (4)ASQ 和 ACQ 寄存器:指示 Admin SQ 和 CQ 基地址;
- (5)SQyTDBL 和 CQyHDBL 寄存器:用于主控端更新相应队列的指针位置。

3 基于 FPGA 的 NVMe 主控端 IP 核

3.1 设计原理

利用 FPGA 的逻辑资源实现 NVMe 协议的高速通信,主要包括用户命令解析、指令执行、数据传输以及系统管理。依据 NVMe 协议 1.3a,系统原理框图如图 2。

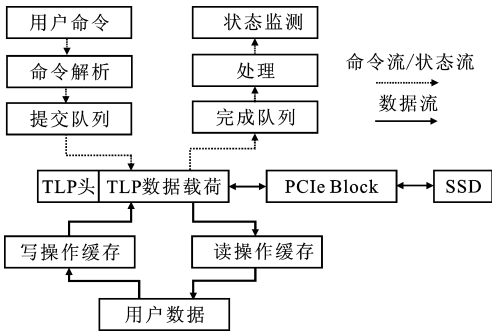


图 2 NVMe 主控端 IP 核原理框图

收到用户命令后首先解码为 64 Byte 的 NVMe 命令码,并存入提交队列中;指令执行与数据传输均通过 PCIe 存储器读写操作进行,因此主控端需完成 PCIe TLP 包的封装以及解封,将数据和命令码发送到 PCIe Block 集成块,并将收到的 TLP 包解封为数据和完成包,放入相应的数据缓存区和 CQ 中。Xilinx 公司在器件中提供了集成硬件模块 PCIe Block 的解决方案,满足本文需求,因此本文调用该资源进行功能整合,并采用 128 bit-Tx/Rx AXI4-Stream 总线进行数据读写和命令传输。

3.2 系统架构

根据上述原理,本文采用模块化理念完成对主控端 IP 核架构设计,根据图 1 所示的命令处理流程,可将该 IP 核分为 NVMe_Ctr、Queue_pkg、Data_Buf、Data_Ctr 四个子模块进行 VHDL 设计,其中设计图如图 3 所示。

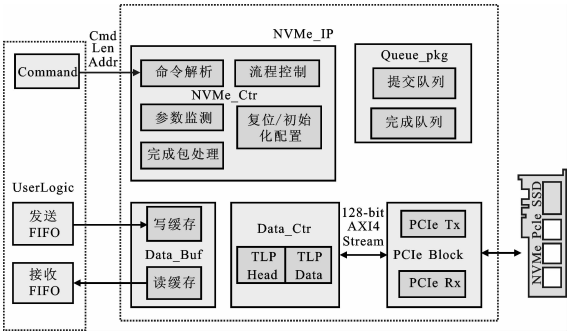


图 3 NVMe 主控端 IP 核模块化设计

3.2.1 NVMe_Ctr 模块

该模块作为整个主控端 IP 的控制中心,主要完成用户命令解析、SQ 尾部/CQ 头部控制、指令操作流程控制、状态参数监测、初始化配置、系统复位以及对完成包的处理。

考虑 FPGA 内存资源并简化寻址机制,当用户命令长度大于两个内存页大小时,自动拆解为多条命令。因此,除最后一条命令外,其它命令的数据传输尺寸为两个内存页。当完成命令解码后,该模块将 64 Byte 命令码发送至 Queue_pkg 模块提交队列尾部,并保存每一个 SQ 尾部指针以通知设备端需处理的新命令个数。

该模块包含控制指令操作流程的状态机,根据图 1 所示的指令执行流程,共分为 8 个执行状态,状态进入后向 Data_Ctr 模块发送使能信号,指令执行完毕后,所有信号自动复位为初始值,进入 IDLE 状态,以待接收后续命令。

3.2.2 Queue_pkg 模块

作为主控端的提交队列和完成队列,该模块由伪双端口 Block RAM 实现,用于存放解码后的 64 Byte 命令码和指令执行后收到的 16 Byte 完成包。当第一个 SQ 为 Full 后,自动存入下一个 SQ,直至所有 SQ 存满,停止接收命令。

3.2.3 Data_Buf 模块

该模块是读写用户数据的缓存区,由两个 128 KByte 伪双端口 BlockRAM 组成,数据宽度为 128 bit,深度为 8 K。

3.2.4 Data_Ctr 模块

该模块是 NVMe 主控端与 PCIe Block 的信息交互模块,主要负责 PCIe TLP 的封装与解封,根据命令执行过程,主要包括以下 PCIe TLP 包的处理:

(1)更新 SQ 尾部 DB 寄存器时,把 SQ 尾部地址和相应的 DB 寄存器封装为 Mwr TLP 包发送到 PCIe Block;

(2)设备端控制器取指时,首先解封来自 PCIe Block 的 Mrd TLP 包,并把 SQ 命令封装为 Cpld TLP 包的数据段;

(3)对于 Identify 命令,解封来自 PCIe Block 的 4KByte Mwr TLP 包,去掉包头,仅留取数据段作为提取硬盘剩余存储空间的依据;

(4)对于 write 命令,首先解封来自 PCIe Block 的 Mrd TLP 包,并把用户数据封装为 Cpld TLP 包的数据段;

(5)对于 read 命令,解封来自 PCIe Block 的

Mwr TLP 包,去掉包头,仅留取数据段并传送到读缓存区;

(6)获取 MSI-X 中断信号,并发送至 NVMe_Ctr;

(7)更新 CQ 头部 DB 寄存器时,把 CQ 头部地址和相应的 DB 寄存器封装为 Mwr TLP 包发送到 PCIe Block。

3.3 操作流程

3.3.1 初始化配置流程

系统上电后自动进入初始化配置阶段,复位信号有效,首先进行 PCIe 初始化,包括配置空间映射、链路状态检查与参数监视,直至 PCIe Block 进入 L0 状态,且 PCIeLinkup 信号有效;PCIe 初始化完成后,进行 NVMe 寄存器配置,配置完成后,IP 核进入 IDLE 状态,需配置的寄存器功能主要包括:

(1)配置内存页大小为 64 KByte;

(2)提交队列位宽 64 Byte,Admin 提交队列深度为 32,I/O 完提交队列深度为 8 KByte,并分配基地址;

(3)完成队列位宽 16 Byte、Admin 完成队列深度为 32,I/O 完成队列深度为 256,并分配基地址;

(4)选择仲裁方式为 RR 仲裁。

3.3.2 指令操作流程

命令操作主要包括 identify、write、read 命令,其中首条命令必须为 identify 命令,识别硬盘的剩余存储空间。当收到用户命令后,IP 核完成命令的解析并解码为 64 Byte 命令码存放于提交队列中。根据 NVMe 协议 1.3a,指令执行过程中,主控端和 SSD 之间将通过多次 PCIe 存储器读写操作。

首先主控端通过 PCIe Memory Write TLP 更新 SQ 尾部 DB 寄存器;设备端控制器通过 DB 寄存器的更新可解析出提交队列中待提取命令的个数,并发送 PCIe Memory Read TLP 读取新命令,主控端收到命令后返回携带 SQ 命令码的完成报文。设备端控制器收到 NVMe SQ 码后解析命令内容:

(1)对于 Identify 命令,通过 PCIe Memory Write TLP 返回 4KByte 数据结构,该数据结构包含硬盘可用存储空间等信息;

(2)对于 write 命令,通过 PCIe Memory Read TLP 读取相应数据,主控端返回的完成报文中会携带用户数据,直至所有数据全部传输;

(3)对于 read 命令,通过 PCIe Memory Write TLP 将数据发送给主控端;

数据传送完成后,设备端控制器通过 PCIe

Memory Write TLP 发送完成包,由主控端 IP 核完成解析并发送至完成队列;随后,设备端控制器发送 MSI-X 中断信号通知主控端 IP 核命令数据传输完成;IP 核收到命令后检查完成队列条目,对成功执行完毕的完成包直接丢弃,未成功的完成包重新进行数据传送,直至完成队列为空,通过 PCIe Memory Write TLP 更新 CQ 头部 DB 寄存器。

4 仿真验证

IP 核收到用户命令后,Busy 信号置高,经 PCIeTx 端口发送更新 DB 寄存器、64 Byte SQ 命令码和用户数据,图 4、图 5 的仿真以 Identify(返回 4 K 数据包)、Write 128 KByte、Read 128 KByte 数据为例;经 PCIeRx 端口接收设备端控制器返回的读取 SQ 头部命令、读用户数据命令、完成包和中断信号。

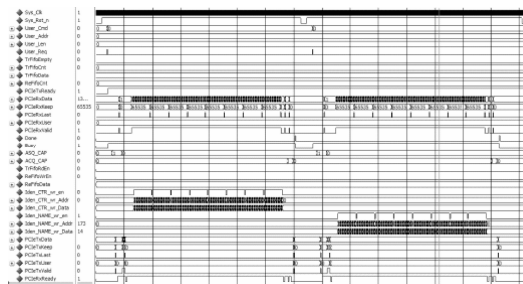


图 4 Identify 命令仿真测试

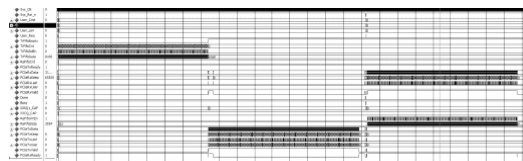


图 5 Write 128KByte、Read 128KByte 命令仿真测试

5 结束语

当前市场下,SATA SSD 已略显疲态,随着市

场存量的不断消解,最终将被 NVMe/PCIe 产品所取代.在航天应用领域,FPGA 以其丰富的逻辑资源与可靠性保证一直被作为主控系统的首选方案.因此,结合 FPGA 的并行加速特性,本文设计了基于 FPGA 的 NVMe 的主控端 IP 核,集成度高、扩展性强、运用灵活方便,特别适合需要大容量存储的嵌入式系统,具有良好嵌入式应用前景.后续该 IP 核还可以进一步拓展应用于 RAID 阵列,设计并实现更高性能的存储技术。

参考文献:

- [1] 董星.基于 FPGA 的存储控制器及相关系统设计技术研究[D].杭州:浙江大学,2016.
- [2] NVMe 标准组织. NVM_Express_Revision_1.3a[EB/OL].(2017-10-24)[2018-08-24] <http://www.nvmexpress.org/about/>.
- [3] SSD FANS.深入浅出 SSD:固态存储核心技术、原理与实战[M].北京:机械工业出版社,2018.
- [4] 王齐.PCIe 体系结构导读[M].北京:机械工业出版社,2010.
- [5] 盘勇军,黄伯铮. VxWorks6.8 操作系统下的 NVMe 驱动设计[J].航空电子技术,2017,48(4):32-33.
- [6] 王磊.基于 VPX 的嵌入式高性能存储技术研究[D].杭州:浙江大学,2018.
- [7] 贾效玲,张刚,张博.SATA 硬盘控制器的 FPGA 实现[J].火力与指挥控制,2016,41(9):182.

作者简介:

崔丹丹 女,(1990-),硕士研究生.研究方向为空间数据存储. E-mail:cuidandan16@csu.ac.cn.

宫永生 男,(1984-),博士,副研究员.研究方向为空间数据存储.

(上接第 25 页)

- [6] GOLDSTEIN H, POOPLE C, SAFKO J, et al. Classical Mechanics, 3rd ed. [J]. American Journal of Physics, 2002, 70(7):782-783.
- [7] TANYILDIZI E, DEMIR G. Golden Sine Algorithm: A Novel Math-Inspired Algorithm[J]. Advances in Electrical & Computer Engineering, 2017, 17(2): 71-78.

作者简介:

肖子雅 女,(1994-),硕士研究生.研究方向为智能算法,项目调度与优化,项目管理.

刘升(通讯作者) 男,(1966-),博士,教授.研究方向为智能计算,机器学习.

E-mail: ml8817568572_1@163.com.